CUDA implementation of Parallel Ford-Fulkerson Algorithm for Maximum Flow Problem

CS16B011, CS16B029, CS16B107

April 7, 2019

1 Introduction

A flow network or transportation network is a weighted directed graph in which edges represents a pathway for the flow and the weight represents the capacity of the link. The flow originates from a source and drains at the sink. At all other nodes the incoming flow must equal the outgoing flow. A flow network can be used to model traffic in computer networks, currents in electric circuits, fluids transportation etc.

Lester R. Ford and Delbert R. Fulkerson proposed the Ford-Fulkerson algorithm, a greedy algorithm to compute the maximum flow in a flow/transportation network. The core of the algorithm is to keep updating the flow in the network as long as it finds augmenting paths. We are going to implement a parallel version of the Ford-Fulkerson Algorithm [1] in CUDA.

2 Maximum Flow Problem Definition

Given a directed graph **G = (V,E)** with source **s** and sink **t**. The directed graph **G** is assumed to have no self loops, only one directed edge between a pair of nodes, no incoming edges to the source and no outgoing edges from the sink. The **capacity** of the edge **c(u,v)** represents the maximum flow through the edge **(u,v)**. The flow from u to v denoted by **f(u,v)** should satisfy the following two constraints:

1. $\mathbf{f(u,v) \leq c(u,v)}$, for each **(u,v)** ∈ **E**.   Capacity constraint.

2. ∀ **v** ≠ **s,t** $\sum_{(u,v) \in E} \mathbf{f(u,v)} = \sum_{(v,w) \in E} \mathbf{f(v,w)}$.   Flow Conservation.

The net flow F through the network is defined as:

$$F = \sum_{(s,v) \in E} \mathbf{f(s,v)} = \sum_{(v,t) \in E} \mathbf{f(v,t)} \tag{1}$$

(i.e) The net outgoing flow from the source or the net incoming flow to the sink.

Given the input flow network, the goal of the problem is to find the maximum flow f satisfying these constraints with minimum runtime.

3 Available Resources

[1] discusses a version of the Ford-Fulkerson algorithm which can be parallelized. It also gives some details as to how we could go about parallelizing the algorithm. They have also mentioned about their simulation results and how much speedup they have got for various kinds of testbeds. We are planning to refer the paper for ideas on parallelization and to compare our speedup with what they have achieved.

Ford-Fulkerson algorithm being a fairly well known algorithm has a lot open source serial implementations which we are planning to use to verify the correctness of our implementation.

There is also a parallel implementation of the algorithm on GitHub. But since it doesn't use the same version of Ford-Fulkerson that we are planning to implement there isn't much to refer to in their implementation. The GitHub repository maintains a residual graph data structure whereas we are following a labelling based implementation.

4 Initial ideas on parallelization

Consider the part of algorithm where an augmenting path is to be found. Let a node be called labelled if the max flow which can pass through it (potential) is known and let it be called scanned if all it's neighboring nodes are also labelled.

In the serial version while scanning a node all it's neighbors which are unlabelled are updated sequentially. The idea is do the updates parallelly for not just a particular node but all nodes which are labelled and unscanned.

But in the above method, the pairs of nodes which are labelled and unlabelled respectively have to computed sequentially. This process can be avoided by going through all edges parallelly and checking for the above condition before updating the potential. This way even the limiting sequential step can be eliminated. Hence the implementation can potentially be sped up by the total number of edges.

Since there are few other parts to the algorithm, they can be tried to be done parallelly using CUDA streams. Concepts such as tiling might be difficult to introduce due to the non geometric nature of the input graph. Also we have to find efficient ways of using the shared memory per block for larger graphs.

References

[1] Zhipeng Jiang, Xiaodong Hu, and Suixiang Gao. A parallel ford-fulkerson algorithm for maximum flow problem. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, page 70. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2013.